Amendments to the Claims

The listing of claims will replace all prior versions and listing of claims in the application:

Listing of claims:

This listing of claims will replace all prior versions and listings of claims in the application:

1. (Currently Amended) A method for compiling a user's logic design for implementation into a functional verification system such that communication bandwidth is increased between the functional verification system and a host workstation by allowing for ~~to allow~~ greater read and write access ~~for reading and writing~~ to memories and registers in the user's logic design, comprising;

identifying all of the memories and registers in the user's logic design;

synthesizing accessibility logic into the user's logic design such that the user's logic design is supplemented by said accessibility logic, said accessibility logic creating access ports to the memories and registers in the user's logic desgin, the access ports facilitating writing of data received from the host workstation to the memories and registers in the user's logic design, said access ports further facilitating reading of data stored in the memories and registers in the user's logic design for transfer to the host workstation.

2. (Currently Amended) The method of claim 1 further comprising the step of assigning a unique identifier to each of the memories and registers in the user's logic design.

3. (Currently Amended) The method of claim 2 wherein said accessibility logic comprises selecting logic, said selecting logic adapted to receive said unique identifier and select a particular one of the memories and registers in the user's logic design.

4. (Currently Amended) The method of claim 3 wherein said accessibility logic comprises logic to read from or write to said particular one of the memories and registers in the user's logic design.

5. (Currently Amended) The method of claim 4 wherein said accessibility logic comprises decode logic that receives commands from a host and controls execution of reading and writing data to the memories and registers in the user's logic design.

6. (Currently Amended) A hardware-assisted design verification system for verifying a target logic circuit design, said verification system having a host workstation in communication with a hardware accelerator, the target logic circuit design comprising Boolean logic gates, registers and memories, the host workstation loading data to or unloading data from the registers and memories, comprising:

protocol logic synthesized into the target logic circuit design, said protocol logic comprising:

an incoming packet register in communication with said host workstation, said incoming packet register storing packets that include data and commands communicated from the host workstation;

an outgoing packet register in communication with said host workstation, said outgoing packet register storing packets that include data to be communicated to the host workstation;

command decode logic, said command decode logic decoding ~~a~~ said commands in said incoming packet register to identify a particular operation, register or memory location in said target logic circuit design, where said particular operation can comprise a write command and where said particular operation can also comprise a read command;

write command execution logic to write data stored in said incoming packet register into said register or memory location in said target logic circuit design for a write command decoded at said command decode logic;

read command execution logic to read data from said register or memory location in said target logic circuit design and store said data in said outgoing packet register for a read command decoded at said command decode logic; and

interface logic interfacing said registers and memories in said target logic circuit design.

7. (Original) The hardware-assisted design verification system of claim 6, wherein said protocol logic includes logic to determine whether data from said incoming packet register is new and control activation of command decoding and execution.

8. (Cancelled).

9. (Cancelled).

10. (Currently Amended) ~~The method of claim 9, further comprising the steps of:~~ A method of synthesizing a packet-based protocol logic for providing access to registers and memories in a target logic circuit design when performing functional verification using a hardware accelerator, comprising:

determining fixed sizes of a request packet, said request packet comprising tag, command, and data end fields;

counting how many of the registers are present in the target logic circuit design;

counting how many of the memories are present in the target logic circuit design;

determining a maximum identification field size of said request packet;

determining a maximum number of data bits of the registers in the target logic circuit design;

determining a maximum number of data bits of the memories in the target logic circuit design;

determining a maximum number of address bits of the memories in the target logic circuit design;

determining a maximum number of bits to send the register data, memory data, and memory address to the target logic circuit design to determine data field size of said request packet

creating an incoming packet register coupled to an input data buffer in the hardware accelerator;

creating an outgoing packet register coupled to an output data buffer in the hardware accelerator;

creating a command decode block to decode a command in said incoming packet register;

creating an execution logic to execute a command decoded at said decode block; and

creating interface logic to access the registers and memories in said target logic circuit design.

creating a memory identification register to identify the memories in the target logic circuit design;

creating a memory address register to provide a current memory address for access;

incrementing said current memory after a memory read command or a memory write command is executed;

creating a finite state machine to indicate that the packet-based protocol logic is in either non-memory mode, continuous memory write mode, or continuous memory read mode; and

creating a state transition control that selects said non-memory mode when said continuous memory operation ends, said state transition control further selecting said continuous memory write mode when said continuous memory write operation is initiated, said state transition control further selecting said continuous memory read mode when said continuous memory read operation is initiated.